Strategies for solving index one DAE with non-negative constraints: Application to liquid-liquid extraction

Ludovic Métivier, Philippe Montarnal

Strategies for solving index one DAE with non-negative constraints: Application to liquid-liquid extraction

Ludovic Métivier[1,*], Philippe Montarnal[1]

CEA Saclay, Gif-sur-Yvette, France

Abstract

Liquid-liquid extraction modeling leads to solve an index one DAE system. For the sake of robustness, it is desirable to account for non-negative constraints. Based on the DASSL architecture (a classical index one DAE solver) we propose and compare three different strategies to implement these bound constraints. Each of these strategies corresponds to a different Newton modification: clipping, damping, or interior point method. The comparisons are made on two test cases: the Robertson ODE problem, and an example from liquid-liquid extraction modeling.

Keywords: differential algebraic system, non-negative constraints, variable stepsize and order BDF methods, nonlinear solver, conservation of mass, trust-regions, constrained dogleg methods
2000 MSC: 34, 49, 68

1. Introduction

The context of the work presented here is the study of liquid-liquid extraction modeling [1] [25]. This industrial process is applied in numerous fields, such as ore processing, oil refining, or nuclear wastes recycling. It is based mainly on the transport of two immiscible liquid phases (generally an aqueous and an organic phase) in opposite directions inside extractors. Depending on the affinities of species for one or the other phase, separation of components can be efficiently achieved, using one or more cycles of liquid-liquid extraction.

The model associated with this process belongs to the general class of reactive transport models: multiphasic flows are transported within an extractor, and different chemical reactions are defined inside each phase, as well as mass transfer between the different phases. The corresponding model is expressed in terms of PDAE (Partial Differential Algebraic Equations), involving transport operators (such as the convection-diffusion operator), and local chemical

*Corresponding Author
Email address: ludovic.metivier@ujf-grenoble.fr (Ludovic Métivier)
[1]CEA Saclay, DEN/DANS/DM2S/SFME/LSET

operators. Recent developments have shown that efficient numerical methods based on the method of lines [20] can be derived for these problems: a semi-discretization in space is performed, yielding a system of Ordinary Differential Equations (ODE) or more generally Differential Algebraic Equations (DAE) [6]. Appropriate solvers are then used to solve the problem.

In the case of liquid-liquid extraction modeling, the semi-discretization in space yields an index one DAE problem[1]:

$$My'(t) = f(t, y(t)), \tag{1}$$

where $y(t) \in \mathbb{R}^m$ is the solution, and $y'(t) \in \mathbb{R}^m$ its derivative in time. In addition, $M \in \mathbb{M}_m(\mathbb{R})$ is a constant matrix possibly singular. When M is non singular, the DAE system is reduced to a standard ODE (Ordinary Differential Equations) system.

As other DAE problems coming from different fields of physics, chemistry, and biology (see for instance the testset for IVP solvers [16]), the liquid-liquid extraction problem is subject to non-negativity constraints

$$\forall t, \quad y(t) \geq 0, \tag{2}$$

since the components of $y(t)$ correspond either to a quantity of moles or chemical concentrations.

Satisfying these constraints is *crucial*. Computing negative components of the solution $y(t)$ is not satisfactory from a physical point of view. Moreover, $f(t, y(t))$ may not be defined for negative values of the components of $y(t)$, and this could prevent converging to the final solution.

Classical DAE solvers can be separated into two distinct families:

- those based on Runge-Kutta implicit time discretization;
- those based on linear multistep implicit time discretization.

Implicit Runge-Kutta methods are unconditionnally stable, for all orders. However, their computation costs tend to increase rapidly as the order grows. Indeed, the size of the nonlinear systems yielded by the implicit discretization is multiplied by the number of stage of the Runge-Kutta method. Among the numerous implicit Runge-Kutta methods, consider for instance the Radau IIA methods. These methods are particularly interesting because of their inherent stability[1]. The order p of these methods verify $p = 2s-1$, where s is the number of stages required by the method. An implementation of this method of fifth order thus require 3 stages. This implies solving a $3 \times m$ nonlinear system at each iteration. This can be an important drawback when attacking large scale problems, even if some techniques can be employed to reduce the computational cost of the algorithm.

[1] For more details about the theory of DAE and the definition of their index, the reader is referred to [3]

[1] These methods are L-stable [9]

On the other hand, the range of stability of linear multistep methods is bounded, but these methods require considerably less computational efforts. A well known example of linear multistep methods is the BDF (Backward Differentiation Formula) method, which is A-stable up to order 2, and $A(\alpha)$-stable up to order 6 [9]. This method only requires solving an m nonlinear system at each iteration.

Among several DAE solvers, such as Radau5 [9] or PSIDE [11], based on Runge-Kutta methods, MEBDFDAE or MEBDFI, based on modified extended BDF methods [5] [19], and more recently BiMD, based on Blended Implicit Methods [4], or GAMD [12], based on the Generalized Adams Methods[2] we focus on DASSL [3]. This solver belongs to the second category mentioned above: it is a variable stepsize and order DAE solver based on a BDF time discretization implemented in a prediction-correction framework. The stepsize and order selection algorithm is particularly efficient. DASSL also implements a non-negativity constraint algorithm in the form of a clipping method. Roughly, this amounts to setting to 0 the negative components of the solution at each time step. This rather simple algorithm is confronted with three main difficulties:

- it does not conserve the total mass in the considered system;
- in certain cases, the clipping method can induce a large number of time steps and reduce the efficiency of the method;
- it does not prevent generating intermediate iterates with negative components, possibly outside the definition domain of $f(t, y(t))$.

Other algorithms for implementing non-negativity constraints have been proposed in the context of ODE. Shampine et al propose a redefinition of the ODE outside the feasible region, that allows the constraint to be followed when the solution reach the bound [23]. Although interesting, this approach can not be applied to DAE, essentially because the equations defining the system cannot be redefined in the context of DAE. A second possibility, proposed by Sandu [21], consists in computing at each iteration the solution of the problem without bound constraints and projecting it into the feasible set. This method, however, is designed only for solving chemical kinetic systems without transport operators or algebraic constraints. Finally, another algorithm is proposed by Gobbert et al. in the context of implicit time-discretization methods such as the BDF method [7]. At each iteration, the nonlinear system is solved by a damped Newton method: the Newton steps computed are multiplied by a damping factor in order to maintain the sequence of iterates within the feasible region. This algorithm, contrary to the previous one, can easily be adapted for solving index one DAE systems.

Another possibility for accounting for non-negativity constraints consists in replacing the classical Newton method used at each time iteration by an inte-

[2]Blended Implicit Methods and Generalized Adams methods are designed to combine advantages from both Runge-Kutta and linear multistep based methods.

rior point Newton method. We focus particularly on the CODOSOL method [2]. This method is based on the trust-region strategy, and uses a system of projection of the descent direction generated at each iteration in order to maintain the sequence of iterates within the feasible region. Although more sophisticated than the damping method, this method requires greater computational efforts. Conversely, it can handle arbitrary lower and upper bound constraints, whereas extending the damping method to this situation could be more difficult.

This work is dedicated to a comparison of the damping and CODOSOL strategies, based on the DASSL framework: BDF time discretization method, stepsize and order selection algorithm, convergencet tests. The clipping method is used as a reference method to perform the comparisons. Two test cases are considered: the classical Robertson test case, which is a stiff ODE problem [23], and a test case derived from liquid-liquid extraction modeling.

In Section 2, we briefly review the principle of DASSL. In Section 3, we describe in more details the three different strategies for satisfying the non-negativity conditions: clipping, Newton damping, and the CODOSOL algorithm. In Section 4, we first propose a comparison on a canonical test case: the Robertson problem. Then, we introduce the liquid-liquid extraction model and compare the three different strategies on a particular test case. Conclusions are given in Section 5.

2. DASSL basis

2.1. BDF discretization

DASSL uses a BDF discretization of the time derivatives $y'(t)$ of the DAE system (1). This method is used under its variable stepsize and order fixed leading coefficient formulation [9]. The implementation is made through a prediction-correction algorithm.

Let $t_n \in \mathbb{R}$ be the time reached at iteration $n \in \mathbb{N}$, and $k \in \mathbb{N}$ the order of the BDF method used to approximate $y(t_n)$. At time t_n, $k+1$ approximations of the solution $y(t_{n-i}), i = 0, \ldots, k$, denoted by $y_{n-i}, i = 0, \ldots, k$ have been computed.

At this stage, the method aims at computing an approximation of the solution at time t_{n+1}, namely $y(t_{n+1})$. We denote this approximation by y_{n+1}. The stepsize h_{n+1} is defined as

$$h_{n+1} = t_{n+1} - t_n. \tag{3}$$

A first approximation of the solution is computed through the definition of the predictor polynomial $w_{n+1}^P(t)$, such that

$$w_{n+1}^P(t_{n-i}) = y_{n-i}, \quad i = 0, \ldots, k. \tag{4}$$

The predictor polynomial interpolates the $k+1$ previous approximations of the solutions at times $t_{n-i}, i = 0, \ldots, k$. Therefore, $w_{n+1}^P(t)$ is of order k. Let

$y_{n+1}^{(0)}$ be the prediction of the solution at time t_{n+1}, and $y_{n+1}^{\prime(0)}$ its time derivative such that

$$y_{n+1}^{(0)} = w_{n+1}^{P}(t_{n+1}), \;\; y_{n+1}^{\prime(0)} = w_{n+1}^{\prime P}(t_{n+1}). \tag{5}$$

The final approximation y_{n+1} is computed through the corrector formula. This formula is the fixed leading coefficient form of the k^{th} order variable stepsize BDF method. The solution of this formula is such that the corrector polynomial and its time derivative satisfy the DAE (1) at t_{n+1}. In addition, the corrector polynomial interpolates the predictor polynomial at k previous points equally spaced, distant from h_{n+1}.

$$\begin{cases} w_{n+1}^{C}(t_{n+1}) = y_{n+1} \\ w_{n+1}^{C}(t_{n+1} - ih_{n+1}) = w_{n+1}^{P}(t_{n+1} - ih_{n+1}), \;\; i = 1, \ldots, k \\ G(t_{n+1}, w_{n+1}^{C}(t_{n+1}), w_{n+1}^{\prime C}(t_{n+1})) = 0. \end{cases} \tag{6}$$

Using the properties of the predictor and the corrector polynomial, an expression can be derived for the time derivatives y'_{n+1} depending on y_{n+1}, $y_{n+1}^{(0)}$ and $y_{n+1}^{\prime(0)}$

$$y'_{n+1} = y_{n+1}^{\prime(0)} - \frac{\alpha_s}{h_{n+1}} \left(y_{n+1} - y_{n+1}^{(0)} \right), \tag{7}$$

where α_s is the fixed leading coefficient of the BDF method

$$\alpha_s = -\sum_{j=1}^{k} \frac{1}{j}. \tag{8}$$

More details on how this expression is obtained are given in [3], based on the work of Jackson and Sack-Davis [13].

Using (7), computing y_{n+1} amounts to solving the nonlinear system

$$G\left(t_{n+1}, y_{n+1}, y_{n+1}^{\prime(0)} - \frac{\alpha_s}{h_{n+1}} \left(y_{n+1} - y_{n+1}^{(0)} \right) \right) = 0. \tag{9}$$

Define the functional $F_{n+1}(y)$ by

$$\begin{array}{rccc} F_{n+1}: & y & \longrightarrow & F_{n+1}(y) = G\left(t_{n+1}, y, y_{n+1}^{\prime(0)} - \frac{\alpha_s}{h_{n+1}} \left(y - y_{n+1}^{(0)} \right) \right) \\ & \mathbb{R}^m & \longrightarrow & \mathbb{R}^m. \end{array} \tag{10}$$

Solving the DAE system (1) using DASSL amounts to solving a sequence of nonlinear systems such that

$$F_{n+1}(y) = 0, \;\; n \in \mathbb{N} \tag{11}$$

2.2. Stepsize and order selection algorithm

What makes DASSL very efficient is its stepsize and order selection algorithm which can be decomposed into two steps:

- the first consists in deciding whether the approximation given by the solution of (11) is accurate enough, or should be rejected;
- the second consists in determining the order and the stepsize for the next iteration.

2.2.1. Selecting or rejecting the approximation y_{n+1}

The selection algorithm is simply based on an approximation of the truncature error. This error can be expressed in terms of the norm of the difference between the predicted value and the solution of (11) [3]. Therefore, the error test implemented in DASSL is

$$C|y_{n+1} - y_{n+1}^{(0)}| \leq 1, \tag{12}$$

where $|.|$ is the norm used by DASSL, taking into account the absolute and relative tolerance criterion $(atol, rtol) \in \mathbb{R}^m \times \mathbb{R}^m$ defined for each component of the solution

$$|v| = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(\frac{v_i}{rtol_i|(y_n)_i| + atol_i}\right)^2}, \quad \forall v \in \mathbb{R}^m, \tag{13}$$

and C depends only on the previous timesteps

$$C = \sum_{j=1}^{k+1}\frac{h_{n+1}}{h_{n+1} + h_n + \ldots + h_{n+2-j}} - \sum_{j=1}^{k}\frac{1}{j}. \tag{14}$$

This implies that the predicted solution should not be too far from the solution of (11). On the other hand, the factor C reflects the effect of the order and stepsize variations on the error test:

- C decreases as the order increases, which allows a less accurate prediction for high order steps;
- for a fixed order, C increases as the stepsize increases; a more accurate prediction is thus required when taking larger steps;
- conversely, for a fixed order, C decreases as the stepsize decreases; a smaller step thus allows a less accurate prediction.

2.2.2. Selecting the order and the stepsize

The order selection is based on the algorithm proposed by Shampine and Gordon [24]. It is set up on the following principle: let T_{k-2}, T_{k-1}, T_k, T_{k+1} denote the estimations of the truncature error which would have been obtained at the current step for orders $k-2, k-1, k, k+1$ respectively. Their detailed expressions can be found in [3]. If

$$T_{k-2} < T_{k-1} < T_k < T_{k+1}, \tag{15}$$

then the order can be increased. Indeed, if the error could decrease at a higher order, it is advantageous to use a higher discretization method. Conversely, if this is not the case, the solution probably varies rapidly, and the polynomial approximation using the former approximations $y_n, \ldots, y_{n-k}$ is no longer reliable. Then the order should be decreased.

In practice, the selection algorithm implemented in DASSL starts by testing if the order should be decreased, whether the approximation y_{n+1} has been accepted or rejected at the former stage of the algorithm. Then only if the approximation has been accepted and it is necessary to decrease the order is the possibility of increasing the order investigated.

Next, a scaling factor r for the time step is computed, related to the chosen order $\overline{k}$, such that

$$r = \left(\frac{2T_{\overline{k}}}{\overline{k}+1} \right)^{-1/(\overline{k}+1)} \tag{16}$$

If r is larger than or equal to 2 and the step has been accepted, then the time step is doubled. If this is not the case, the time step remains constant, or is decreased, depending on the indicator value. Note that the factor r increases if the error decreases, which makes sense: if the error is small, the time step should be increased.

DASSL code uses a maximal order of 5 [3]. Although BDF methods are α-stable until order 6, this value should be avoided for robustness [9].

This strategy provides a method that aims at taking timesteps as large as possible. When the solution varies slowly, high order discretization is stable enough to be used, thus the order of the BDF method is increased, and the stepsize can be large. Conversely, when the solution starts to present fast variations, high order methods require small timesteps to remain stable. Hence, it is worthwhile to first decrease the order, before decreasing the stepsize. Indeed, if the solution varies rapidly, the former approximations of the solution does not provide valuable information on the current approximation by the polynomial interpolation. Therefore, low order methods are more stable than high order ones. Choosing these low order methods yields the possibility of taking larger steps, which improves the efficiency of the algorithm in terms of computation time. This rather complex selection algorithm is at the core of DASSL and makes the method very efficient.

2.3. Solving nonlinear systems

Solving the nonlinear system (11) with a Newton method amounts to generating a sequence y^q from an initial guess y^0 such that

$$y^{q+1} = y^q - J(y^q)^{-1} F_{n+1}(y^q), \tag{17}$$

where $J(y^q)$ denotes the Jacobian matrix of $F_{n+1}(y^q)$;

$$J(y^q) = F'_{n+1}(y^q). \tag{18}$$

This requires solving the linear system

$$J(y^q)p^q = -F(y^q), \tag{19}$$

which is done through an LU factorization of $J(y^q)$ [8].

In DASSL, the initial guess y^0 is the prediction y^0_{n+1} given by the predictor polynomial. Moreover, DASSL implements a modified version of the Newton

algorithm, designed especially to save computation time. The most expensive part of the integration scheme is the evaluation of the Jacobian matrix and its LU factorization. The number of entries is the square of the number of unknowns. Therefore, instead of recomputing the Jacobian matrix $J(y^q)$ at each iteration of the Newton method, DASSL keeps the same approximation of the Jacobian $J(y^0)$ throughout the nonlinear iterations, and stores its LU factorization. This approximation is not systematically recomputed at each time step. Indeed, for small time steps, the Jacobian changes are negligible, and the same approximation can be used. Based on this prinicple, a special indicator is computed in the original implementation of DASSL, to determine when the Jacobian has to be recomputed. In our implementation, however, we decided to simplify the algorithm and systematically recompute the Jacobian whenever the timestep or the discretization order changes, as proposed in [7]. In our numerical experiments, we observed that this srategy only slightly increases the overall number of Jacobian computations compared to the original DASSL algorithm.

2.4. Stopping criterion

The stopping criterion for the Newton method is chosen as a combination of the classical Newton methods and DASSL stopping criteria. The convergence is declared whenever one of these two conditions is met. The Newton method convergence condition corresponds to

$$\|F_{n+1}(y)\| \leq atol_N, \tag{20}$$

where $atol_N \in \mathbb{R}$ is an absolute tolerance criterion defined for the Newton method. This means that the current iterate is close enough to the solution of the nonlinear system. The DASSL condition is different. It can be expressed as

$$\frac{\rho}{1-\rho}|y^{q+1} - y^q| < 0.33, \tag{21}$$

where ρ reflects the convergence rate of the Newton method

$$\rho = \left(\frac{|y^{q+1} - y^q|}{|y^1 - y^0|}\right)^{1/q}. \tag{22}$$

Near the solution, the convergence should be quadratic, in which case ρ decreases rapidly. Then the factor $\frac{\rho}{1-\rho}$ also decreases and the iterations stop even if the distance between the current and the previous iterate is large. On the other hand, if ρ does not decrease rapidly, the iterations stop when the distance between the current and the previous iterate becomes small enough. This is to be understood as a linear convergence condition.

In addition to these two tests, a maximum permissisble number of Newton iterations is introduced. This number is usually set to 4 in DASSL (as is in the ODE15s solver of the ODE suite in Matlab [22]).

When Newton iterations do not converge, the stepsize is reduced by a factor of one quarter. A new nonlinear system is thus defined through the time discretization algorithm described in section 2.1, based on the new stepsize. If the stepsize becomes smaller than a fixed tolerance criterion, CDASSL stops and returns an error flag.

Based on this algorithm, we present in the next section three different strategies for introducing non-negativity constraints.

3. Three different strategies for satisfying non-negativity constraints

3.1. Clipping method

The clipping method is the simplest strategy which can be proposed to ensure non-negativity. Consider the solution y_{n+1} of (11). For a given threshold $\eta > 0$, if any component of y_{n+1} is lower than $-\eta$, the solution y_{n+1} is refused. Then the time step h_{n+1} is shortened, and a new nonlinear system is built. If not, any negative component of y_{n+1} is between $-\eta$ and 0. All these components are set arbitrarily to 0.

The same algorithm is applied in case of a non feasible initial guess. If some components of the prediction y_{n+1}^0 are smaller than $-\eta$, then the initial guess is replaced by the previous accepted step y_n (which is in the feasible region). If not, the prediction is kept as initial guess, but all its negative components are set to 0.

This simple method has three main drawbacks.

- The mass conservation of the system is violated by forcing some components of the solution of the nonlinear system to 0.
- The sequence of iterates generated by the Newton method will not necessary remain within the feasible set, thus the method does not prevent from trying to evaluate the function $f(t, y(t))$ outside its definition domain.
- The method induces time step reduction and recomputation when some components become lower than $-\eta$. Depending on the threshold value, one of these drawbacks becomes prominent: if η is small, then the number of refused steps can be large; conversely if η is large, the number of refused steps is small, but the error on mass conservation increases.

Therefore, we are looking for a more robust strategy, that ensures mass convervation without excessive computation costs, and generating a sequence of iterates within the feasible region.

3.2. Newton-damping method

The second method is the damping method proposed by Gobbert et al [7]. Consider the Newton method presented in Section 2.3. A natural way of assuring the non-negativity of the solution y_{n+1} is to ensure that at each iteration the

Newton update does not produce an iterate which lies outside the feasible region. This can be achieved by computing a damping factor α^q such that

$$\left\{ \begin{array}{l} \alpha^q = \min\limits_{i=1,\ldots,m} \alpha_i^q \quad \text{where} \\ \\ \alpha_i^q = \left| \begin{array}{l} 1 \text{ if } (y_{n+1})_i^q + p_i^q \geq 0 \\ -\dfrac{1}{p_i^q}\left((y_{n+1})_i^q + \varepsilon\right) \text{ if } (y_{n+1})_i^q + p_i^q < 0, \end{array} \right. \end{array} \right. \tag{23}$$

where ε is a threshold factor that ensures the damping factor α^q never vanishes. Then the update formula (17) is replaced by

$$y^{q+1} = y^q + \alpha^q p^q. \tag{24}$$

At this stage, some components of the solution of a nonlinear iteration possibly lie between $-\varepsilon$ and 0. Thus, the strategy is complemented by a projection of y^{q+1} into the feasible set.

The initial guess is also modified to satisfy the non-negativity constraints: if some components of the prediction are negative, then another initial guess is tested, namely

$$y^0 = y_n + (y_n - y_{n-1}). \tag{25}$$

If this initial guess still presents some negative components, then the previously described damping strategy is applied to produce an initial guess in the feasible region, replacing p^q by $y_n - y_{n-1}$ and y^q by y_n.

This produces an elegant and efficient strategy, ensuring all the iterates stay within the feasible region. The method is therefore robust. In addition, mass conservation is also satisfied up to the threshold value ε. Indeed, the magnitude of component of the solution projected into the feasible set is never larger ε. The user can tune this parameter according to the tolerance required on mass conservation.

3.3. The CODOSOL method

We first give the general description of a trust-region based Newton method. Then we briefly describe how CODOSOL adapts this method to handle bound constraints [2].

3.3.1. Trust-region method

Solving (11) with a Newton method can be recast as solving a sequence of quadratic problems

$$\min_p G(p) = \frac{1}{2}\|J(y^q)p - F_{n+1}(y^q)\|^2, \;\; y \geq 0. \tag{26}$$

Indeed, the solution of (26) is given by cancellation of the gradient of the quadratic function $G(p)$ which is

$$\nabla G(p) = J(y^q)^T J(y^q)p - J(y^q)^T F_{n+1}(y^q). \tag{27}$$

Provided $J(y^q)$ is nonsingular,

$$\nabla G(p) = 0 \Longleftrightarrow p = -J(y^q)^{-1} F_{n+1}(y^q). \tag{28}$$

This is also equivalent to solving the optimization problem

$$\min_y f(y) = \frac{1}{2}\|F_{n+1}(y)\|^2, \tag{29}$$

by successive quadratic approximations of the cost function $f(y)$.

The trust-region method consists in defining a region of radius Δ^q in which the quadratic approximation $G(p)$ is minimized. At each nonlinear iteration, the following problem is solved

$$\min_p G(p), \quad \|y^q + p\|^2 \leq \Delta^q. \tag{30}$$

This subproblem is solved with the following procedure:

- 1. First, compute the Newton solution $p^N = -J(y^q)^{-1} F_{n+1}(y^q)$.
- 2. If $\|p^N\|^2 \leq \Delta^q$ select the Newton step: $p = p^N$, and go directly to step 5.
- 3. If $\|p^N\|^2 > \Delta^q$, compute the Cauchy step :

  $$p^C = -\tau J(y^q)^T F_{n+1}(y^q) = -\tau \nabla f(y^q), \tag{31}$$

  where τ is computed to minimize $f(y^q)$ along the steepest descent direction $\nabla f(y^q)$ within the trust region.
- 4. Compute $\gamma = \min_\gamma G(p(\gamma))$ where $p(\gamma) = p^N + (1 - \gamma p^C)$.
- 5. Form the new iterate $y^{q+1} = y^q + p(\gamma)$.
- 6. Compare the accordance rate ρ between the quadratic approximation and the actual cost function $f(y)$. While $\rho < \beta_1$, for a given parameter β_1, refuse step $p(\gamma)$, shrink the trust-region radius, and restart at step 3. If $\rho > \beta_1$, select $p(\gamma)$. If $\rho > \beta_2$, for a given parameter $\beta_2 > \beta_1$, increase the trust-region radius Δ^q.

The trust-region algorithm was originally developed as a globalization of the Newton method[1]. Indeed, it is well-known that if the initial guess y^0 is too far from the solution, the Newton method may not converge. Thus, the trust-region strategy can be interpreted as follows: if the norm of the classical Newton step is small enough for the next iterate to stay within the trust-region, then the Newton step is chosen, and the method does not differ from the usual Newton method. However, if the Newton step produces an iterate outside the

[1]This globalization strategy is an alternative to the more usual linesearch method.

trust region, the step actually taken is a convex combination of the Newton step and the steepest descent step. If the accordance between the current quadratic approximation of the misfit function and the actual misfit function is good, then this step can be followed. If it is not the case, the trust-region is shrunk and the step is rejected.

While the Newton method may not converge if the initial guess is too far from the solution, the steepest descent method converges from any starting point (possibly at a very slow rate). The trust-region method proposes a compromise between these two methods, represented by the coefficient γ. Far from the solution, the trust-regions are small, and the influence of the steepest descent is stronger (γ is small). When the algorithm approaches the solution, the trust-region radius increases, the quadratic approximation of the misfit function is improved, and the influence of the Newton direction increases (γ is larger). The resulting method converges globally, with a local quadratic convergence rate [2].

In terms of computational effort, the trust-region method only requires one linear system to be solved at each iteration to compute the Newton step. However, the evaluation of the accordance rate ρ between the quadratic approximation and the actual cost function requires one additional function evaluation. Therefore, the trust region method requires more computational efforts than the classical Newton method.

3.3.2. Accounting for bound constraints

Based on the trust-region method, only a few modifications are necessary to handle bound constraints. These modifications are based on the same considerations as for the Newton damping method: basically, it is necessary to modify the step taken by the algorithm to produce iterates that stay within the feasible region.

This is done in the CODOSOL method. First, the Newton direction p^N computed at step one of the step selection algorithm is projected into the feasible set. Second, the Cauchy step is also modified following an algorithm sharing similarities with the damping method presented in subsection 3.2. Then appropriate formula can be derived to compute the coefficient τ and γ. A complete description of the method is beyond the scope of this paper, and the reader is referred to [2] for more details.

3.3.3. Initial guess

We complete the CODOSOL method with a suitable strategy to deal with initial guesses that do not lie within the feasible region. Indeed, the CODOSOL method is supposed to start from a feasible initial guess. We chose to implement the strategy proposed by Gobbert et al. for the damping Newton method [7]. Note that this strategy amounts to selecting a step closer to the previous solution y_n at time step $n+1$. This is consistent with the fact that y_n, as the last step computed, is certain to be in the feasible region, whereas no guarantees are available about the prediction y^0_{n+1}.

4. Numerical results

Based on the DASSL architecture, we compare below the three strategies for imposing non-negativity on two test cases. The first test case is the classical Robertson problem. This is an ODE system derived from kinetic chemistry, involving three unknowns and three kinetic reactions. The discrepancy between the orders of magnitude of the reaction rates is significant, yielding a very stiff problem, and possible blow up of the solution for negative component of the solution. This test case has been studied by numerous authors [23] [7] and is thus a good starting point to compare the three strategies.

The second test case derives from our liquid-liquid extraction application. This model involves a larger number of unknowns (318), and also presents strong discrepancies between the characteristic times of the different physical phenomena, yielding a stiff problem, that is also more realistic with regard to potential future applications.

4.1. The Robertson problem

The Robertson problem can be stated as follows

$$\left\{\begin{array}{lcrr} \partial_t y_1 & = & -0.04y_1 + 10^4 y_2 y_3 & \\ \partial_t y_2 & = & 0.04y_1 - 10^4 y_2 y_3 & -3\times 10^7 y_2^2 \\ \partial_t y_3 & = & & 3\times 10^7 y_2^2, \end{array}\right. \tag{32}$$

with initial conditions

$$y_1(0) = 1, \quad y_2(0) = 0, \quad y_3(0) = 0. \tag{33}$$

The corresponding concentrations profiles from $t = 0$ to $t_f = 4 \times 10^{11}$ are given in Figure 1. The integration time t_f is chosen large enough to ensure the solution reach stability. The results shown in Figure 1 are obtained using the CODOSOL strategy, but clipping and damping strategies give equivalent plots.

An interesting feature of the Robertson problem is that the total mass of the system should be conserved throughout the iterations. Indeed, the solution of (32) satisfies

$$\forall t, \;\; y_1(t) + y_2(t) + y_3(t) = 1. \tag{34}$$

This provides an efficient mass conservation indicator MC for the different strategies:

$$\mathrm{MC} = \max_{t\in[t_0;t_f]} |y_1(t) + y_2(t) + y_3(t) - 1|. \tag{35}$$

We found that the solution computed by the DASSL algorithm, using the classical Newton iterations without bound constraints does not blow up as presented in [7], even with very coarse tolerance parameters. In addition, for restrictive tolerance parameters ($atol = 10^{-6}$ and $rtol = 10^{-6}$ for instance), the solution does not contain any negative components. This difference probably comes from the fact that the experiments in [7] are based on the ODE15s code of the Matlab ODE suite [22], whereas this work is based on the DASSL architecture.

	Nsteps	Nfailures	Nfeval	Njeval	MC	Nclipping
Clipping	224	15	381	162	3.34×10^{-8}	2
Damping	224	15	381	162	1.01×10^{-12}	N/A
CODOSOL	220	14	645	158	2.43×10^{-12}	N/A

Table 1: Comparison between clipping, damping and CODOSOL methods for the Robertson test case

Therefore, we choose tolerance parameters for which the solution contains negative components when non enforcement of non-negativity is used. This makes is possible to compare our different strategies. This leads us to the standard tolerance parameters of the ODE suite [22] :

$$atol = 10^{-6}, \quad rtol = 10^{-3}. \tag{36}$$

In addition, the tolerance parameter on the Newton algorithm is set equal to $atol$:

$$atol_N = atol. \tag{37}$$

The maximum permissible number of Newton iterations is set to 4. The damping parameter ε is set to 10^{-12}. For the clipping method, we tried several values for the clipping parameter η. It turns out that choosing a value too restrictive in order to preserve mass conservation precludes the convergence. Thus, we finally chose the smallest value for which we obtained convergence, $\eta = 10^{-7}$.

The performance of the different strategies in terms of the order chosen, number of Newton iterations at each time step, and the chosen time steps are equivalent. The results obtained with CODOSOL method are shown in Figure 1. The same plots are obtained with the clipping and the damping methods.

For each strategy, we report the number of time steps taken (Nsteps), the number of time steps refused (Nfailures), the total number of function evaluations (Nfeval), the total number of Jacobian evaluations (NJeval), the mass conservation indicator MC, and the number of components set to 0 (Nclipping). The results are indicated in Table 1.

Not surprisingly, the error in mass conservation method is larger for the clipping method than for the damping and CODOSOL methods. Note that only 2 clippings are performed. This means that the solution computed by DASSL for these tolerance parameters without non-negativity constraints is almos non-negative. However, 2 clippings are sufficient to introduce artificial mass in the system and cause an error larger by 4 orders of magnitude than for the two other methods.

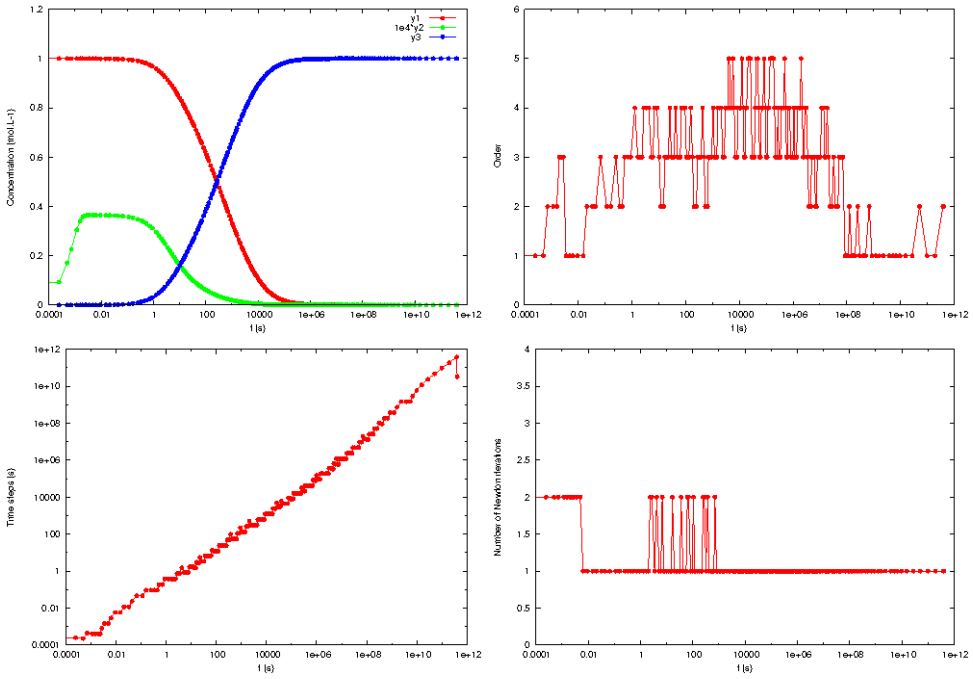

Figure 1: Concentration profiles obtained using the CODOSOL strategy (top left), order chosen as a function of time (top right), time step chosen as a function of time (bottom left), number of Newton iterations per time steps (bottom right)

The errors in mass conservation caused by the damping and CODOSOL methods are of the same order. In terms of computation, the number of steps and failures is almost the same for the three strategies, even if it is slightly smaller for the CODOSOL method. In turn, the number of evaluations of the function is far more important for the CODOSOL method. This is not surprising, since the trust-region strategy requires additional evaluations of F in its step selection procedure, in order to update the size of the trust-region. Finally, the number of evaluations of the Jacobian is the same for the clipping and the damping method, and slightly lower for the CODOSOL method.

Note that the strategy applied to deal with unfeasible initial guesses is very important. Simply clipping the prediction, without checking the magnitudes of its negative components, can lead to substantial mass errors. This was observed in particular for the CODOSOL method. Conversely, choosing the previous accepted state modified with an order one upgrade, (possibly damped to remain in the feasible region) turns out to be efficient.

We conclude from the study of the Robertson test case that the clipping method is inefficient compared to the damping method and the CODOSOL method. These two methods give equivalent results, except in terms of the number of evaluations of the function F. This comes directly from the trust-region strategy, which requires at least one additional evaluation at each iteration. However, this test case only involve 3 unknowns, and the constraints are almost inactive (only 6 components are set to 0 throughout the iterations using the clipping method). Therefore we are interested in testing these strategies on a test case that is more representative of the problems encountered in liquid-liquid extraction modeling.

4.2. The liquid-liquid extraction test case

4.2.1. Modeling

We summarize here the equations describing the liquid-liquid extraction problem. More details about the construction of the model can be found in [17].

The liquid-liquid extraction process is based on the transport of an organic and an aqueous phase in opposite directions, inside an extractor. An emulsion of the two immiscible phases is generated by the extractor to activate mass transfer between them. This emulsion is generally described in terms of the phase that remains continuous (the continuous phase) and the one that is dispersed in droplets in the continuous phase (the dispersed phase). In the sequel the organic phase is continuous and the aqueous phase is dispersed.

Chemical system. In our example, the aqueous phase contains six species, namely

$$A, B, D, F, G, H, \tag{38}$$

whereas the organic phase contains five species, namely

$$E, BE, HE, DE, FE. \tag{39}$$

The process modeled is the extraction of component B from the aqueous phase using the extractor E from the organic phase to form the species BE in the organic phase. This is performed using an organic phase containing the components E and DE. The extraction efficiency of B involves a strong concentration of H (an acid for example). Unfortunately, component DE of the organic phase is back-extracted into the aqueous phase in the form of component D which reacts with H to form component F, which degrades the efficiency of the extraction of B. To mitigate this difficulty, a component A is introduced in the aqueous phase, to react with D and form a component G. This is modeled by the following reactions.

In the aqueous phase, two kinetically-controlled reactions are defined

$$D + F \longrightarrow H \quad (aq1), \quad A + D \longrightarrow G \quad (aq2) \tag{40}$$

For each of these reactions, a reaction rate is defined

$$v_{aq1} = k_{aq1}[D][H], \quad v_{aq2} = k_{aq2}[A][D], \tag{41}$$

with

$$k_{aq1} = 10^2, \;\; k_{aq2} = 10^7. \tag{42}$$

Four kinetic reactions are defined to model the transfer between the organic and the aqueous phase:

$$\left\{ \begin{array}{rcll} B + E & \longrightarrow & BE & (trans1) \\ DE & \longrightarrow & D + E & (trans2) \\ H + E & \longrightarrow & HE & (trans3) \\ F + E & \longrightarrow & FE & (trans4). \end{array} \right. \tag{43}$$

The corresponding reaction rates are:

$$\left\{ \begin{array}{ll} v_{trans1} = k_{trans1}[B][E][H]^{\alpha} & v_{trans2} = k_{trans2}[DE][E] \\ v_{trans3} = k_{trans3}[H][E] & v_{trans4} = k_{trans4}[F][E], \end{array} \right. \tag{44}$$

with

$$k_{trans_1} = 1, \;\; k_{trans_2} = 0.1, \;\; k_{trans_3} = 0.001, \;\; k_{trans_4} = 1, \;\; \alpha = 1.9 \tag{45}$$

Transport. The chemical species are transported in a simple mixer-settler extractor. This extractor comprises successive stages. Each stage is composed of one biphasic mixer and one biphasic settler. The aqueous flow is introduced at the last stage, while the organic flow is introduced at the first stage. Each mixer flows into the settler of the same stage. The aqueous phase of a settler flows into the mixer of the previous stage. The organic phase of a settler flows into the mixer of the next stage. This produces a countercurrent circulation of the two phases inside the extractor (see Figure 2).

From a numerical point of view, each mixer and each phase of the settlers is represented as an elementary volume p. The total number of elementary volumes is $P = 3 \times N_s$ where N_s is the number of stages of the mixer/settler.

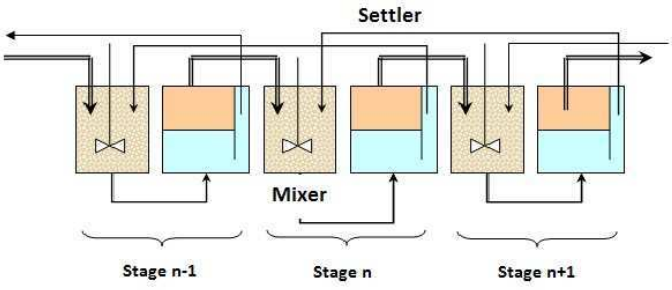

Figure 2: Mixer/settler description

We assume that the hydraulic part of the system is at its steady state: the flows between the elementary volumes are equal to the flows introduced into the mixer-settler. Two outflows are also defined at the extreme settlers. The aqueous phase of the first settler exits from the mixer-settler, as well as the organic phase of the last settler. These outflows are also equal to the flows that are introduced in the mixer-settler. We denote by d^{aq} and d^{org} the organic flow and the aqueous flow respectively.

The volumes of each phase in the mixers are computed given the total volume V_i^M, $i = 1, \ldots, N_s$ of the mixers and the assumption of well stirred flow. For an arbitrary stage i, we have

$$\begin{cases} V_i^{aq} + V_i^{org} = V_i^M \\ \dfrac{V_i^{aq}}{V_i^{org}} = \dfrac{d^{aq}}{d^{org}}, \end{cases} \tag{46}$$

where V_i^{aq} (respectively V_i^{org}) denotes the volume of the aqueous phase (respectively the organic phase) in the mixer at stage i.

The situation is simpler for the settlers, for which we assume the interface between the two phases is fixed. The user can thus decide what are the volumes of each phase in the settlers.

We order the elementary volumes in the following way: at each stage,

- the first elementary volume is the aqueous phase of the settler;
- the second elementary volume is the mixer;
- the third elementary volume is the organic phase of the settler.

Using this ordering of the elementary volumes, the transport in the mixer/settler

is described by linear operators T_p^{aq} and T_p^{org} such that, for $n = 1, \ldots, N_s$

$$\begin{cases} T_p^{aq}(m^{aq}) = d^{aq}(m_{p+1}^{aq} - m_p^{aq}), & p = 3n & \text{(settler aqueous phase)} \\ T_p^{aq}(m^{aq}) = d^{aq}(m_{p+2}^{aq} - m_p^{aq}), & p = 3n+1 & \text{(mixer aqueous phase)} \\ T_p^{org}(m^{org}) = d^{org}(m_{p-2}^{org} - m_p^{org}), & p = 3n+1 & \text{(mixer organic phase)} \\ T_p^{org}(m^{org}) = d^{org}(m_{p-1}^{org} - m_p^{org}), & p = 3n+2 & \text{(settler organic phase)} \end{cases} \tag{47}$$

where m_p^{aq} (respectively m_p^{org}) denotes a number of moles of an arbitrary species in the aqueous phase (respectively in the organic phase) of the elementary volume p.

Mass transfer. We use a double layer description of mass transfer [15]. This description assumes the presence of a zone located at the interface between the continuous phase and the dispersed phase (the interfacial zone). The species belonging to each phase diffuse in this zone through a film [18]. These two films express the resistance of the species in their passage from one phase to the other. Corresponding diffusion coefficients K^{aq} and K^{org} are defined.

The interfacial zone has an infinitely small thickness. Hence, it is assimilated to a surface where no accumulation of mass is allowed. This implies that at each time step, the quantity of matter exchanged in the interfacial zone is equal to the quantity that has diffused from the inner phase.

The corresponding surface area is denoted by σ. This surface area depends on the hydraulic properties of the dispersed phase, notably on the mean size of the droplets diameters, for example the Sauter diameter. The amount of exchanged mass between two phases is proportional to this surface.

Equations. The equations are written for a number of moles. These quantities are denoted by the letter of the corresponding species. The volume concentrations are denoted by [.]. Interfacial concentrations are denoted by the superscript

i. For each volume p, the following set of equations is defined:

$$\left\{\begin{array}{rcll}
\partial_t A_p & = & T_p^{aq}(A) & -v_{aq2} \\
\partial_t B_p & = & T_p^{aq}(B) & +K^{aq}\sigma(B_p^i - [B]_p) \\
\partial_t D_p & = & T_p^{aq}(D) & +K^{aq}\sigma(D_p^i - [D]_p) - v_{aq1} - v_{aq2} \\
\partial_t F_p & = & T_p^{aq}(F) & +K^{aq}\sigma(F_p^i - [F]_p) - v_{aq1} \\
\partial_t G_p & = & T_p^{aq}(G) & +v_{aq2} \\
\partial_t H_p & = & T_p^{aq}(H) & +K^{aq}\sigma(H_p^i - [H]_p) + v_{aq1} \\
0 & = & K^{aq}\sigma([B]_p - B_p^i) & -v_{trans1} \\
0 & = & K^{aq}\sigma([D]_p - D_p^i) & +v_{trans2} \\
0 & = & K^{aq}\sigma([F]_p - F_p^i) & -v_{trans3} \\
0 & = & K^{aq}\sigma([H]_p - H_p^i) & -v_{trans4} \\
\partial_t E_p & = & T_p^{org}(E) & +K^{org}\sigma(E_p^i - [E]_p) \\
\partial_t BE_p & = & T_p^{org}(BE) & +K^{org}\sigma(BE_p^i - [BE]_p) \\
\partial_t DE_p & = & T_p^{org}(DE) & +K^{org}\sigma(DE_p^i - [DE]_p) \\
\partial_t FE_p & = & T_p^{org}(FE) & +K^{org}\sigma(FE_p^i - [FE]_p) \\
\partial_t HE_p & = & T_p^{org}(HE) & +K^{org}\sigma(HE_p^i - [HE]_p) \\
0 & = & K^{org}\sigma([E]_p - E_p^i) & -v_{trans1} + 2v_{trans2} - v_{trans3} - v_{trans4} \\
0 & = & K^{org}\sigma([BE]_p - BE_p^i) & +v_{trans1} \\
0 & = & K^{org}\sigma([DE]_p - DE_p^i) & -v_{trans2} \\
0 & = & K^{org}\sigma([FE]_p - FE_p^i) & +v_{trans3} \\
0 & = & K^{org}\sigma([HE]_p - HE_p^i) & +v_{trans4}
\end{array}\right. \tag{48}$$

4.2.2. Numerical results

In the following numerical example, we consider a mixer-settler composed of 8 stages. The constant describing the diffusion from the inner phases to the interfacial zone K^{aq} and K^{org} is set to 1. The final computation time is defined to ensure the chemical system reaches its steady state: $t_f = 100$ h.

The aqueous flow introduced at stage 8 contains the species A, B, and H with the following concentrations

$$[A] = 0.5\text{mol.L}^{-1}, \;\; [B] = 0.5\text{mol.L}^{-1}, \;\; [H] = 1\text{mol.L}^{-1}. \tag{49}$$

The organic flow introduced at stage 1 contains the species E, and DE, with the following concentrations

$$[E] = 0.5\text{mol.L}^{-1}, \;\; [DE] = 1\text{mol.L}^{-1} \tag{50}$$

The velocity of the two flows is set to 1 L.h^{-1}. The volume of the settlers is set to 2 L, with an interface in the middle, which yields a volume of aqueous phase and organic phase equal to 1 L in each settler. The total volume of a mixer is set to 1 L, which yields volumes of aqueous phase and organic phase equal to 0.5 L in each mixer, given that the incoming flow velocities are equal.

At $t = 0$, the initial concentrations are set to 0, except for species A, B, H, E and DE, whose initial concentrations correspond to the concentrations of the incident flows.

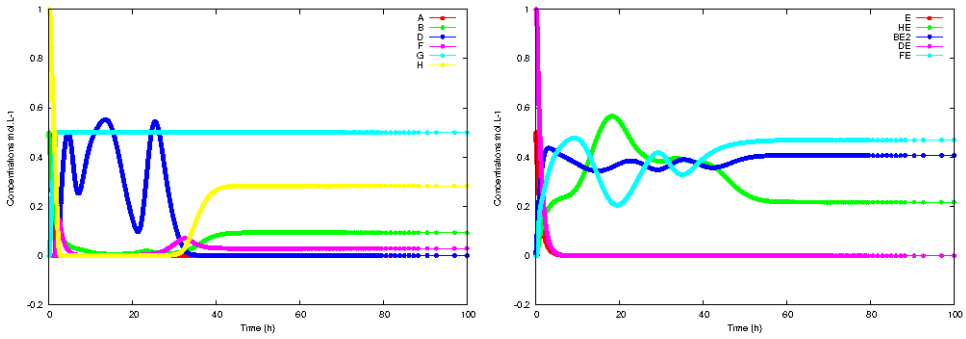

Figure 3: Reference solution obtained with no enforcement and restrictive tolerance parameters $atol = rtol = atol_N = 10^{-10}$. Aqueous phase of the first settler (left), organic phase of the last settler (right).

Because the initial concentrations correspond to the composition of the introduced flows, it is easy to define mass invariants in the system. We consider particularly the invariant corresponding to the number of moles of the species that contain component E:

$$u = E + BE + DE + FE + HE. \tag{51}$$

In each settler the initial quantity of u is equal to 1.5 mol. In each mixer the initial quantity of u is 0.75 mol. Thus, the initial quantity of u in the mixer-settler is $(1.5 + 0.75) \times 8 = 18$ mol. This invariant is an indicator of the mass conserving capacities of the different numerical strategies tested. As for the Robertson case, we define the mass conservation indicator MC such that

$$\text{MC} = \max_{t \in [0;t_f]} |u(t) - 18| \tag{52}$$

We first compute a reference solution using a simple Newton algorithm for the nonlinear iterations and no enforcement of the solution. The tolerance parameters are chosen to be very restrictive, namely

$$atol = rtol = atol_N = 10^{-10} \tag{53}$$

The corresponding results are shown in Figure 3.

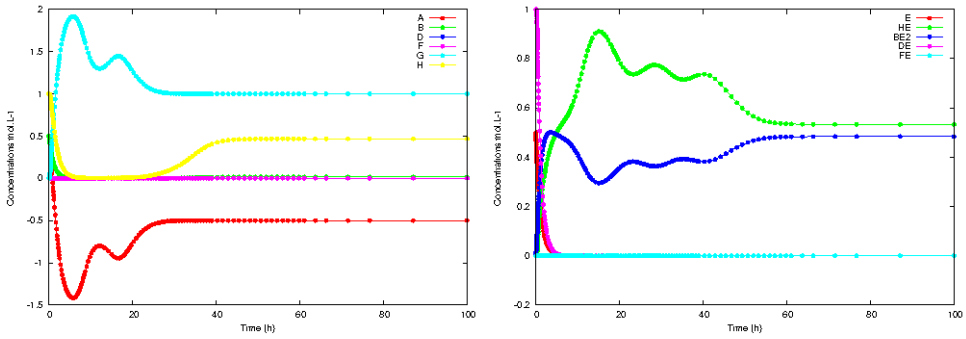

Figure 4: Solution computed without enforcement and tolerance paremeters $atol = rtol = atol_N = 10^{-4}$

We use the same numerical method (DASSL + Newton algorithm without enforcement) with less restrictive tolerance parameters:

$$atol = rtol = atol_N = 10^{-4} \tag{54}$$

The results are indicated in Figure 4. We observe that due to the negativity of A, the solution blows up.

This is a good illustration of the importance of preserving non-negativity for the liquid-liquid extraction problem. On this basis, we compare the three different strategies introduced in section 2.3, using the same tolerance parameters. The corresponding results are presented in Figure 5.

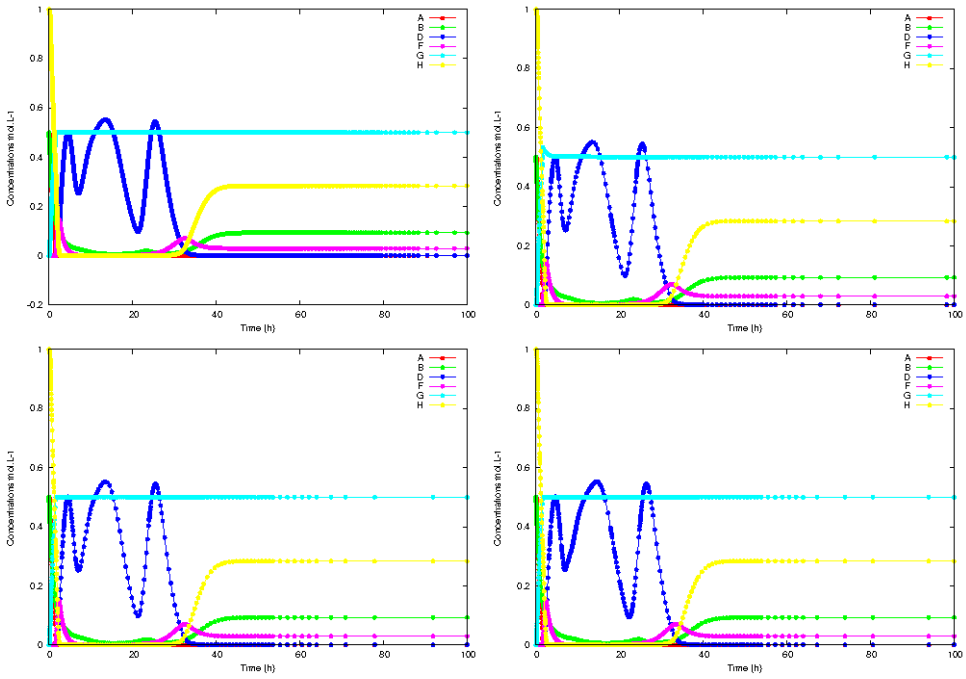

Figure 5: Comparison of the solutions: reference solution (top left), clipping method (top right), damping method (bottom left), CODOSOL method (bottom right).

We tested different values for the η parameter for the clipping method. As for the Robertson case, we observed that too small values prevent the numerical method from converging. Indeed, the number of refused steps increases drastically, and the method finally reduce the time step below its critical value, leading to failure (even for a critical value set to 10^{-50}). We thus choose $\eta = 1$ to obtain the convergence of the clipping method. As a consequence, the error on mass conservation is large, as shown in Table 2. In addition, the solution computed by the clipping method suffers from artifacts compared to the reference solution (see for instance the concentration profile of G in the aqueous phase of the first settler). Before reaching its steady state at 0.5 mol.L^{-1}, the profile grows to 0.55 mol.L^{-1}.

The damping method is used with a damping factor ϵ equal to 10^{-12} as for the Robertson case. The results we obtained are satisfactory. First, the solution does not blow up, as with non enforcement of non-negativity. Second, the convergence is obtained after a few iterations, and the concentration profiles computed respect the reference solution. Third, the mass conservation is ensured.

The CODOSOL method also gives satifsfactory results The computed solution does not blow up. Convergence is obtained in fewer iterations than with the

reference solution. However more iterations are required than for the damping method. The computed concentration profiles do not present numerical artifacts such the ones that can be seen in the results provided by the clipping method, which is also satisfactory. Finally, the conservation of mass provided by the CODOSOL method is very good, the maximum error during the integration of the DAE being equal to 2.13×10^{-14}.

The number of time steps taken (Nsteps), the number of time steps refused (Nfailures), the total number of function evaluations (Nfeval), the total number of Jacobian evaluations (NJeval), the mass conservation indicator MC, and the number of components set to 0 (Nclipping) are reported in Table 2.

	Nsteps	Nfailures	Nfeval	Njeval	MC	Nclipping
Reference	3975	269	13493	1245	8.88×10^{-14}	0
Clipping	375	44	1193	251	9.87×10^{-5}	1069
Damping	343	37	1120	241	4.26×10^{-14}	N/A
CODOSOL	571	96	3405	419	2.13×10^{-14}	N/A

Table 2: Comparison between clipping, damping and CODOSOL for the liquid-liquid extraction test case

The damping method provides the best results in terms of number of steps taken and number of function and Jacobian evaluations. As expected, the mass conservation of the invariant u is very poor for the clipping method: the magnitude of the error on mass is 10^{-4}. Conversely, the damping method and the CODOSOL method provide very good conservation of u, up to the order of 10^{-14}.

The comparison between the CODOSOL method and the damping method also reveals that for this test case, the CODOSOL method is less efficient. Indeed, the number of time steps taken is larger, as the number of Jacobian evaluation. This is also the case for the number of function evalutations, but this is due to the nature of the CODOSOL method, which is based on a trust-region strategy.

In order to better understand why the CODOSOL method seems to be less efficient than the damping method, we compare other aspects of the different strategies. First, we observe the values of the order taken at each step. The corresponding results are shown in Figure 6.

The order evolutions generated by the clipping method and the damping method are almost the same. Conversely, it appears that the order selected at each time

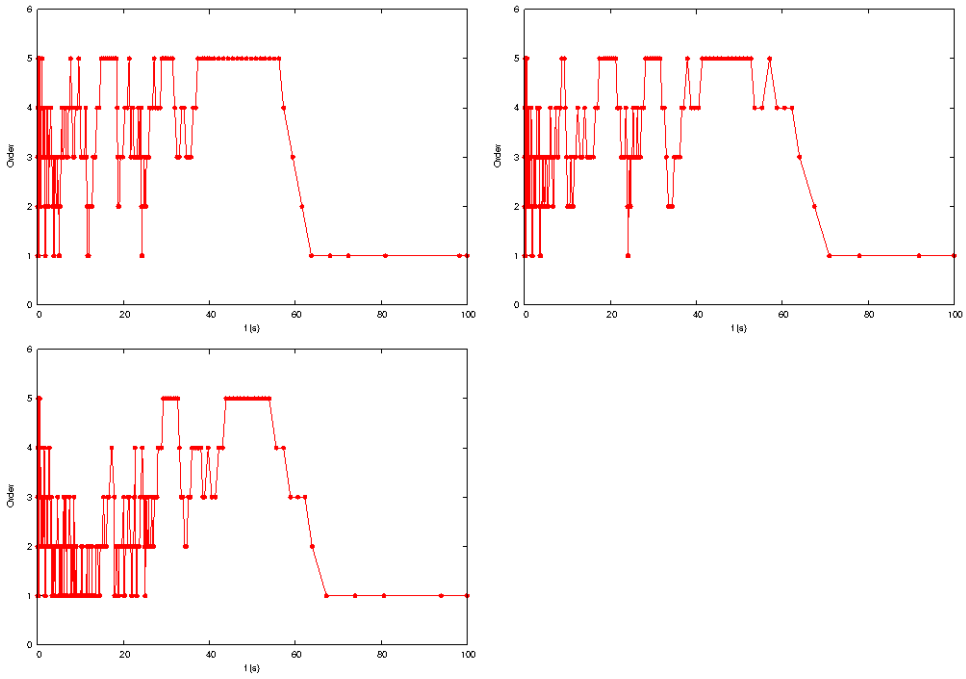

Figure 6: Comparison of the order taken at each time step: clipping method (top left), damping method (top right), CODOSOL method (bottom left)

step by the DASSL method is lower when using the CODOSOL method, at least for the first iterations.

Second, we compare the number of Newton iterations per time step. The results are indicated in Figure 7. We observe the very similar behavior of the clipping method and the damping method. Most of the time, only two iterations of the Newton algorithm are performed. Only for the last iterations, the maximum of 4 iterations is reached. The CODOSOL method exhibits the same behavior with a slight difference for the last iterations: the maximum number of 4 iterations is reached a more often.

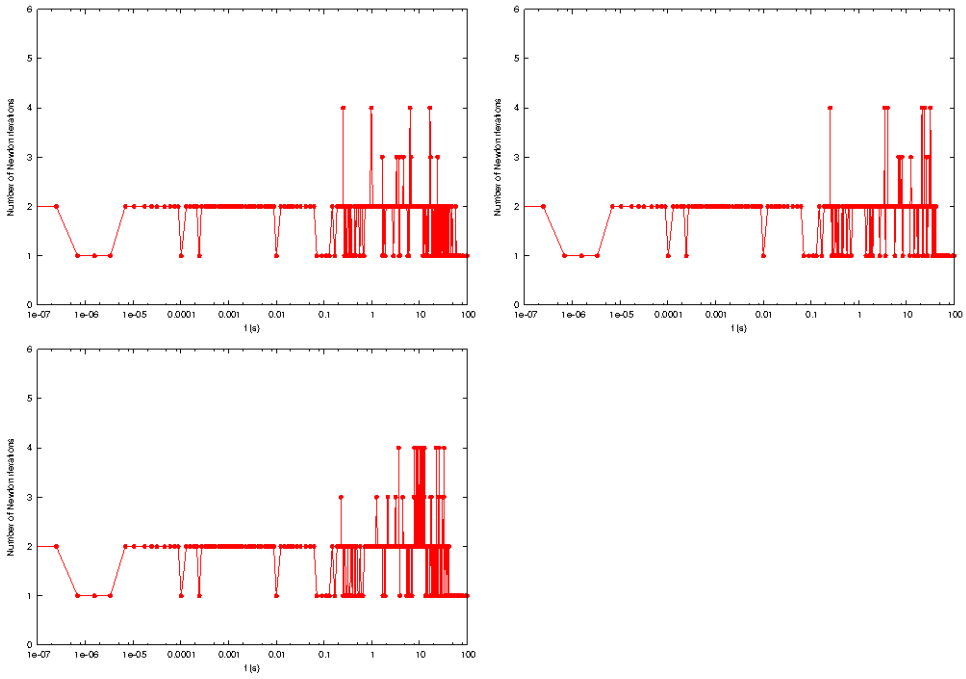

Figure 7: Comparison of the number of Newton iteration at each time step: clipping method (top left), damping method (top right), CODOSOL method (bottom left).

Finally, we compare the size of the time steps taken at each iteration. The results are shown in Figure 8. Once again, the clipping method and the damping method give very comparable results. Conversely, the CODOSOL method seems to encounter more difficulties for the last iterations, which is correlated with the number of Newton iterations performed at these steps. The time step appears to be more irregular on the final iterations, and decreases more often.

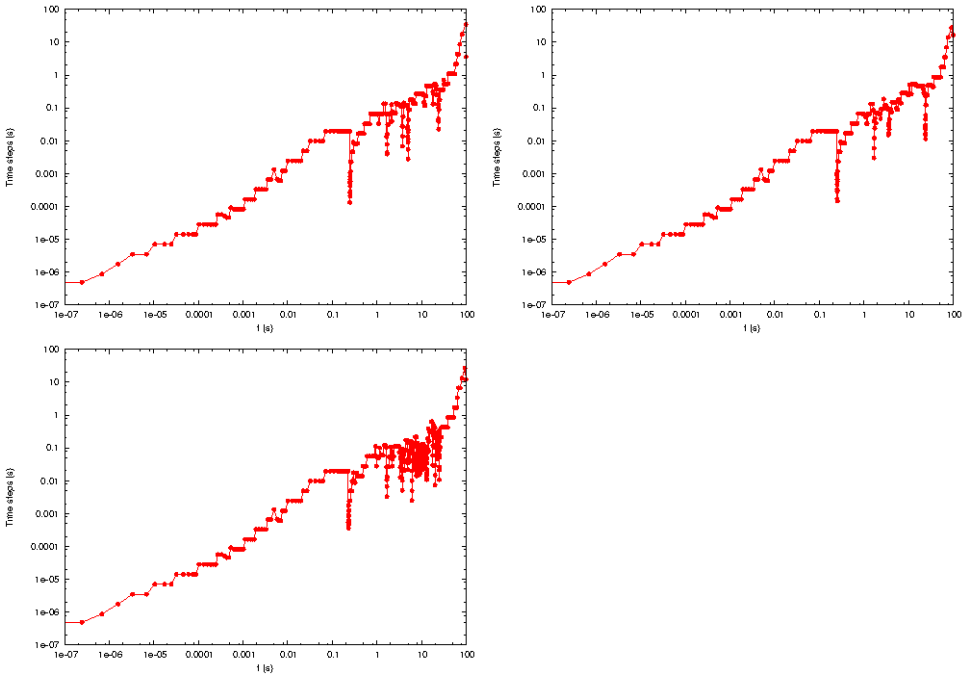

Figure 8: Comparison of the stepsizes: clipping method (top left), damping method (top right), CODOSOL method (bottom left).

The mean values for the order, the number of Newton iterations and the stepsizes are reported in Table 3.

	Mean order	Mean number of Newton iterations	Mean stepsize
Clipping	2.98	1.80	0.37
Damping	2.88	1.83	0.40
CODOSOL	2.05	2.00	0.24

Table 3: Comparison of the mean order, the number of Newton iterations and the mean stepsize

The results displayed in Table 3 confirm what is visible in Figures 6, 7 and 8. The mean values associated with the clipping method and the damping method are quasi-similar. Conversely, the mean order is lower for the CODOSOL method, the mean number of Newton iterations is larger, and the mean stepsize is smaller.

This emphasizes that the CODOSOL method is somewhat less efficient than the damping method for the liquid-liquid extraction test-case.

Two reasons may explain the deterioration of the convergence properties of CODOSOL, compared with the damping method. First, this could come from the method of computation for the modified Newton step inside the CODOSOL algorithm, described in [2]. The modified Newton step always produces an iterate strictly in the feasible region, that may be too far from the boundary compared with the damping method. Another possible explanation comes from the trust-region framework, which may not be suitable for application to DASSL. Indeed, the trust-region procedure is a method that ensures global convergence, that is to say the method converges from any starting point. However in DASSL, the prediction is supposed to be close enough to the solution to allow only for four Newton iterations at each time step.

In conclusion, the results obtained on our DAE test case arising from liquid-liquid extraction show the importance of respecting non-negativity constraints. Indeed, authorizing negative components of the solution can lead to a solution blow up. Among the three different strategies we tested to enforce non-negativity, we illustrate that the clipping method should be avoided. This method, although showing good convergence properties, does not respect mass invariance. In addition, the computed solution may present numerical artifacts. In turns, the damping method and the CODOSOL methods are more effective, regarding mass invariance, and convergence to the reference solution.

The comparison between the damping strategy and the CODOSOL strategy highlights the better convergence properties of the damping method. The mean order, the mean stepsize, and the mean number of Newton iterations are better than with the CODOSOL strategy. The convergence of the CODOSOL strategy is thus slower. In addition, the number of function evaluations is more important, which is due to the trust-region strategy at the core of the CODOSOL method.

However, the CODOSOL method should not be discarded for future applications. Indeed, in some cases, not only non-negativity constraints are necessary, but also upper bound constraints, which may vary with the components of the solution. In this case, the damping method could encounter strong difficulties to converge, as the damping factor is computed for all the components of the solution at the same time. On stiff large-scale problems, in the case of upper and lower bounds; the damping factor allowing to stay within the feasible bounds could be relatively small, which could slow the convergence of the damping method. Conversely, the CODOSOL method is naturally designed to handle complex bound constraints. For this type of applications, the CODOSOL method should prove to be more efficient than the damping method.

5. Conclusion and perspectives

The modeling of liquid-liquid extraction implies solving an index one DAE problem, with non-negativity constraints. While solving index one DAE problems is a well known issue, accounting for non-negativity constraints is more difficult. Non-negativity is however crucial for a large number of applications, mainly because the physical quantitities are intrinsically non-negative (concentrations, number of moles). In addition, negative values of the components of the solution may prevent convergence, either because the function $f(t, y(t))$ defining the DAE system is not defined for these values, or because negative values yield numerical instabilities (as shown for the liquid-liquid extraction test case).

Based on recent work on this subject originally proposed in the context of ODE problems, we present and compare three different strategies for taking these bound constraints into account. The comparisons are based on the architecture of a classical index one DAE solver DASSL. This solver implements a BDF discretization of the time derivatives of the system, in a prediction/correction scheme. It also implements a very efficient stepsize and order selection algorithm. The implicit BDF discretization scheme solves a nonlinear system at each time iteration. This is performed with a Newton solver.

The three strategies for enforcing non-negativity of the solution are variant of the Newton solver. The first one, named as the clipping method, consists in setting to 0 all the components of the solution computed by the classical Newton solver if their absolute exceed a threshold parameter set by the user. If at least one component is lower than this threshold, the solution is rejected, and the time step is reduced, in order to define a new nonlinear system.

The second strategy, known as the damping method, consists in multiplying the Newton step by a damping factor at each nonlinear iteration. This damping factor is designed to force the iterates to stay within the feasible region.

The third strategy, known as the CODOSOL method, consists in replacing the classical Newton method by a more complex Newton method, based on a trust-region globalization, specially designed to handle general bound constraints.

We perform a comparison of these three strategies on two test cases. The first one is the Robertson case, a stiff ODE problem with 3 unknowns considered as a reference case by many authors [23]. This system is stiff, and also provides a mass invariant, making it possible to control the mass-conserving character of the three strategies. The second test case is taken from our liquid-liquid extraction application. The corresponding problem is an index one DAE problem, which is stiff, and involves 318 unknowns. As for the Robertson case, a mass invariant can be computed in order to control the mass conserving character of the different strategies.

The two test cases show that the clipping method should be avoided. Indeed, clipping some components of the solution to 0 produces artificial mass, and destroy the conservation of mass. In addition, nothing prevents attempts to evaluate the function defining the DAE system outside its definition domain,

which may cause the integration to fail. Conversely, both the damping strategy and the CODOSOL strategy provides good mass invariance, and generate sequence of iterates that remain within the feasible region. From the standpoint of a computational, the damping strategy is more effective than the CODOSOL strategy. Because of the trust-region globalization, the CODOSOL method requires larger number of function evaluations. However, this is not as crucial as the number of Jacobian evaluations, which represents the most computation-intensive part of the DAE integration. While for the Robertson test case, the damping and the CODOSOL method give comparable results with regard to this number of evaluations, the damping method outperforms the CODOSOL method on the liquid-liquid extraction test case.

However, it should not be forgotten that the CODOSOL method has at least one advantage over the damping method. The CODOSOL method is able to deal with upper and lower bound constraints possibly differing from one component to another. This seems more difficult to achieve using the damping method. Future perspectives should include comparisons of the two methods on a test case that requires such complex upper and lower bounds. In addition, scaling matrices (as for instance the Coleman-Li matrix[2]) can be used in the CODOSOL method to enhance the handling of bound constraints. This should be also tested in comparison with the damping method.

Acknowledgments

The authors are grateful to S.Bellavia, M.Macconi, S.Pieraccini for their support, interest, and precious advice. They also warmly thank H.Roussel, V.Pacary, C.Balaguer and X.Hérès for fruitful discussions and for their patience in teaching chemistry to applied mathematicians. t

References

[1] M.M.Attarakih, H.J.Bart, T.Steinmetz, M.Dietzen, N.M.Faqir, LLEC-MOD: A Bivariate Population Balance Simulation Tool for Liquid-Liquid Extraction Columns, The Open Chemical Engineering Journal, **2**, 10-34, 2008.

[2] S.Bellavia, M.Macconi, S.Pieraccini, Constrained Dogleg Methods for nonlinear systems with simple bounds, Computational Optimization and Applications, to be published.

[3] K.E.Brenan, S.L.Campbell, L.R.Petzold, Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations, Classics in Applied Mathematics 14, SIAM, North-Holland, New-York, Amsterdam, London, 1995.

[4] L.Brugnano, C.Magherini, Blended Implicit Methods for solving ODE and DAE problems, and their extension for second order problems, J. Comput. Appl. Math., **189**, 34-50, 2006.

[5] J.R.Cash, Efficient numerical methods for the solution of stiff initial-value problems and differential algebraic equations, Proc. R. Soc. Lond.,**459**, 797-815, 2003.

[6] C de Dieuleveut, J.Erhel, M.Kern, A global strategy for solving reactive transport equations, Journal of Computational Physics, **228**, 6395-6410, 2009.

[7] M.K.Gobbert, M.Muscedere, T.I.Seidman, R.J.Spiteri, A non-negativity preserving Newton method for high-order implicit time stepping, submitted.

[8] G.H.Gollub, C.F.Van Loan, Matrix Computation, Third Edition, John Hopkins University Press, Baltimore, MD, USA, 1996.

[9] E.Hairer, S.P.Nørsett, G.Wanner, Solving Ordinary Differential Equations, I, II, Springer, Berlin 1996.

[10] A.C.Hindmarsh, P.N.Brown, K.E.Grant, S.L.Lee, R.Serban, D.E.Shumaker, C.S.Woodward, SUNDIALS: Suite of Nonlinear and Differential/Algebraic Equation Solvers, ACM Transactions on Mathematical Software, **31**, 3, 363-396, 2005.

[11] P.J. van der Houwen, J.J.B. de Swart, Parallel linear system solvers for Runke-Kutta methods, Advances in Computational Mathematics, **7**, 157-181, 1997.

[12] F.Iavernaro, F.Mazzia, Solving ordinary differential equations by generalized Adams Methods: properties and implementation techniques, Applied Numerical Mathematics, **28**, 107-126, 1998.

[13] K.R.Jackson, R.Sacks-Davis, An alternative implementation of variable step size multistep formulas for stiff ODEs, ACM Trans. Math. Software, **6**, 295-318, 1980.

[14] F.T.Krogh, Changing step size in the integration of differential equations using modified divided differences, Proc. Conf. Num. Solution of ODEs, Lecture Notes in Mathematics, **362**, Springer-Verlag, New-York, 1974.

[15] W.K.Lewis, W.G.Whitman, Principles of gas absorption, Ind. Eng. Chem, **16**, p.1215-1220, 1924.

[16] F.Mazzia, C.Magherini, Test set for Initial Value Problem Solvers, report 4/2008, Department of Mathematics, University of Bary, Italy.
http://pitagora.dm.uniba.it/ testset/

[17] L.Métivier, H.Roussel, Accounting robustly for instantaneous chemical equilibriums in reactive transport: a numerical method and its application to liquid-liquid extraction modeling, submitted.

[18] N. Di Miceli Raimondi, "Transfert de matière liquide-liquide en micro-canal : application à la réaction chimique ", PhD Thesis, Toulouse University, France, 2008.

[19] G.Psihoyios, Solving time dependent PDEs via an improved modified extended BDF scheme, Applied Mathematics and Computation, **184**, 104-115, 2007.

[20] W.E.Schiesser, The Numerical Method Of Lines: Integration of Partial Differential Equations, Academic Press, San Diego, 1991.

[21] A.Sandu, Positive Numerical Integration Methods for Chemical Kinetic Systems, Journal of Computational Physics, **170**, 589-602, 2001.

[22] L.F.Sampine, M.W.Reichelt, The Matlab ODE Suite, SIAM Journal on Scientific Computing, **18**, 1-22, 1997.

[23] L.F.Shampine, S.Thompson, J.A.Kierzenka, G.D.Byrne, Non-negative solutions of ODEs, Applied Mathematics and Computation **170**, 556-569, 2005.

[24] L.F.Shampine, M.K.Gordon, Computer Solution of Ordinary Differential Equations, W.H.Freeman and Co., 1975.

[25] O.Weinstein, R.Semiat, D.R.Lewin, Modeling, simulation and control of liquid-liquid extraction columns, Chemical Engineering Science, **53**, 325-339, 1998.